**Pointeri**

Un pointer refera o adresa de memorie … (revedeti lectia in caiet)

…

**// Ex. 1**

int x=7;

int \* px;

px = &x;

cout<<(\*px)<<endl; // 7

(\*px)=8;

cout<<(\*px)<<endl; // 8

cout<<x<<endl; // 8

int \* \* ppx=&px;

cout<<(\*\*ppx)<<endl;

**Pointeri la constante si pointeri constanti**

// **Ex. 2** - **pointer la constanta**

**const** int y=3; // o constanta trebuie initializata OBLIGATORIU la declarare

x=y+3;

cout<<x<<" "<<y<<endl;

// y++; // **EROARE**: y e constanta nu se poate modifica

// int \*py=&y; // EROARE: cannot convert const int\* to int\* (tipul pentru py e int \*, nu e const int \*)

const int \*py = &y; // py este pointer la o constanta

cout<<(\*py)<<endl;

// (\*py)=10; // nu putem schimba o valoare constanta

int a=5;

py=&a; // dar pointerul il pot schimba

a=6; // E OK, a nu e constanta, dar

// (\*py)=6; EROARE: py trateaza valoarea ca o constanta

// **Ex. 3** - **pointer constant**

int z=10;

int w=9;

// int \* const pz; EROARE: pointerul constant trebuie initializat

int \* const pz = &z;

// pz=&w; // EROARE: pz e pointer constant, nu poate retine alta adresa

(\*pz)=20; // E OK, valoarea de la acea adresa nu e constanta

// Deci in acest caz modificarea pointerului nu e permisa (pointer constant) dar continutul referit de pointer se poate modif

/\*

In cazul folosirii unor parametri pointeri, pentru a preveni modificarea conţinutului referit de aceştia se preferă definirea lor ca pointeri la constante. De exemplu o functie care compară două şiruri de caractere are prototipul:

int strcmp(const char \*s, const char \*d);

\*/

**Operaţii aritmetice cu pointeri.**

Asupra pointerilor pot fi efectuate următoarele operaţii:

* adunarea / scăderea unei constante
* incrementarea / decrementarea
* scăderea a doi pointeri de acelaşi tip

Prin incrementarea unui pointer legat de un tip **T**, adresa nu este crescută cu **1**, ci cu valoarea **sizeof(T)** care asigură adresarea următorului obiect de acelaşi tip.

În mod asemănător, **p + n** reprezintă de fapt **p+n\*sizeof(T)** .

Pointerii pot fi comparaţi prin relaţiile **==** şi **!=**  intre ei sau cu constanta simbolică **NULL.**

**Legătura între pointeri şi tablouri**

*În C numele unui tablou este un pointer constant la primul element din tablou:* **x=&x[0]**

Numele de tablouri reprezintă *pointeri constanţi*, deci nu pot fi modificaţi ca pointerii adevăraţi.

Obs. Orice operaţie realizată folosind variabile indexate se poate obţine şi folosind pointeri

Prin urmare *variabilele indexate* pot fi transformate în *expresii cu pointeri* şi avem echivalenţele:

|  |  |  |  |
| --- | --- | --- | --- |
| Adresă | | Valoare | |
| Notaţie indexată | Notaţie cu pointeri | Notaţie indexată | Notaţie cu pointeri |
| **&x[0]** | **x** | **x[0]** | **\*x** |
| **&x[1]** | **x+1** | **x[1]** | **\*(x+1)** |
| **&x[i]** | **x+i** | **x[i]** | **\*(x+i)** |
| **&x[n-1]** | **x+n-1** | **x[n-1]** | **\*(x+n-1)** |

// Ex. 4 - legatura dintre pointeri si tablouri

int b[10]; // numele unui tablou este un pointer constant la primul element din tablou

int \*p;

p=b;

// b=p; // EROARE: b este pointer constant

// b++; // EROARE

b[2]=30;

cout<<\*(p+2)<<endl;

for (int i=0; i<10; i++)

\*(p+i)=i;

for (int i=0; i<10; i++)

cout<<\*(p+i)<<" ";

**Siruri de caractere**

Un sir de caractere se retine in memorie ca un vector ale carui elemente sunt de tip character (char). Sfarsitul sirului de caractere este marcat de o component care retine 0.

Deci daca sirul contine **n** atunci el trebuie sa aiba cel putin **n+1** octeti alocati (fiecare caracter se memoreaza pe un octet + unul pentru sfarsitul de sir, adica pentru 0).

// Ex. 5 - siruri de caractere

char v[]={'a','b','c','\0'}; // sau char c[]={'a','b','c',0};

char s[]="mare";

char cuv[100]="mai mult"; // am rezervat mai multi octeti decat e necesar

char \*c="pointer";//in acest caz c va retine un pointer pornind de la care nu se pot modifica continuturile componentelor

// decat prin functii speciale

cout<<v<<endl;// abc

cout<<s[2]; // r

cout<<s<<endl<<s+1<<endl<<s+2<<endl; //mare are re

s[1]='e';

cout<<s<<endl; //mere

cout<<cuv<<endl;

//c[1]='i'; // nu pot

cout<<c;